Amendments to the Specification

Please replace paragraph [0003]-[0006], [0013] [0015]-[0017], [0019], [0024]-[0026], [0029], [0030], [0032], and [0035] with the following paragraphs respectively:

[0003] It is therefore urgent to require a transmission standard supporting high-speed data transmission. As far as the transmission standard in common use is concerned, IEEE 1394 provides an advanced solution for the high-speed transmission and becomes highly valued. The IEEE 1394 standard is also known as ~~iLink, or FireWire~~ the ILINK or FIREWIRE standard.

[0004] After finishing the design of the interface card with a 1394 controller, the interface card has to be put to the general test and the load flow test to ensure its performance. The test methods presently used are performed under the driver structure in Microsoft's ~~Windows~~ WINDOWS System. The packet of the 1394 standard is generated by the test method and received by the interface card to ensure its normalization.

[0005] However, the drawback of the test method is that it is hard to debug during the load flow test. The load flow test is to send out a large amount of packets to the interface card in a very short time for testing the stability of the interface card. The driver under ~~Windows Operating System~~ the WINDOWS operating system needs an additional debug program, SoftICE of NuMega company for example, to check the values inside the computer memory and the buffer. It is hard to perform debugging for the test worker since the memory address in the Microsoft's WINDOWS System is a virtual address, which requires a transformation for mapping onto the physical address.

It is also hard to trace the wrong packet while problem causing. Besides, it is not easy to monitor the testing status at every moment due to the difficulties of observing the buffer and memory and the inconvenience of changing the buffer setting.

[0006] It is therefore an object of the invention to provide a method for testing interface cards with IEEE 1394 controllers. The method can be performed in a multiple system and it is ~~no need to be~~ not restricted to use in the ~~Windows Operating System~~ WINDOWS operating system .

[0013] This invention is used to put several 1394 controllers to the load flow test. The 1394 controllers are disposed on interface cards individually. The interface cards can be placed in one or several hosts. Referring to FIG. 1A and 1B, two environments to which the preferred embodiment of the invention is applied are illustrated. As shown in FIG. 1A, the test method according to the invention is applied to a single system. The single system includes a host 10 and interface cards 20 and 30 placed in the host 10. The interface card 20 is connected with the interface card 30 by the cable 25. As shown in FIG. 1B, the test method according to the invention is applied to a multiple system. The multiple system includes two systems. One system consists of a host 40 and an interface card 50. The other system consists of a host 70 and an interface card 60. Besides, the interface card 50 is connected with the interface card 60 by the cable 55. The invention is not restricted to be applied to environments as shown in FIG. 1A and FIG. 1B, as it can also be applied to the a multiple environment consisting of several interface cards placed in a host or a multiple environment consisting of several interface cards placed in several hosts.

[0015] The embodiment of the test method is developed in the MS-DOS (Microsoft Disk Operating System) in order to avoid the test method being restricted to use in the ~~Windows Operating System~~ WINDOWS operating system.

[0016] As shown in FIG. 2A to 2D, the flowchart of the test method according to the preferred embodiment of the invention is shown. Referring first to Figure 2A, it starts to initialize the interface cards to be tested and allocate memory in step 210. Since there might be several hosts for building the test environment as shown in FIG. 1B, the operating systems of all the hosts are held in a ~~waited~~ waiting state for being ready in step 215 to proceed with the procedure. In the next step 220, all the interface cards to be tested have to communicate one another for building a test environment. After successfully building the test environment by using all the communication data, one of the interface card is selected as a master interface card and the other interface cards are the slave interface cards. The way to choose the master/ slave interface cards is known by the skilled person in the art and thus omitted without further description. Subsequently, the master interface card and the slave interface cards proceeds to perform different test procedures. As shown in FIG. 2A to 2D, there are dotted line for separating two test procedures. The left one is the test procedure of the master interface card while the right one is the test procedure of the slave interface card. The master interface card and the slave interface card proceed to the master test table and the slave test table respectively and await the instruction from the test worker as shown in step 225 and 230.

[0017] The master test table includes three choices: (1) initializing communication protocol packets; (2) performing the test; (3) end. It enters step 235

when the test worker chooses to initialize communication protocol packets according to the (1) choice in the master test table. It enters M1 node when the test worker chooses to performing the test according to the (2) choice in the master test table. It enters step 240 and concludes the test procedure of the master interface card when the test worker chooses the end according to the (3) choice in the master test table.

[0019] The slave test table includes two choices: (1) performing the test; (2) end. It enters step 245 and concludes the test procedure when the test worker chooses the end according to the (2) choice in the slave test table. It enters S1 node and proceeds with the subsequent steps when the test worker chooses to perform the test according to the (1) choice in the slave test table.

[0024] Referring to FIG. 2C, the slave interface card starts to perform the load flow test after receiving the contact signal from the master interface card. That is, all the interface cards, including the master interface card and the slave interface card, perform the load flow test in step 295. There is a A large amount number of packets are transmitted among interface cards and it waits to complete the command instructed by the packets. For example, the command might be an instruction of reading or writing the data from the memory of some interface card or a host. In the next step 300, the status of every interface card is checked.

[0025] If there is any error has occurred, not only the debugging is performed but also the system needs rebuilding in step 305. After that, in step 306, the master interface card and the slave interface card return to the master test table and the slave test table respectively. As shown in step 310, if there is no error has occurred, the

master interface card sends check packets to the slave interface cards for the preparation of checking the test results, such as the transmission speed of packets. In step 310, the master interface card proceeds to the M2 node and the slave interface card proceeds to the S2 node.

[0026] Referring to FIG. 2D, in step 315, the slave interface card sends the third ready signal to the master interface card for responding to the check packets. In step 320, the master interface card detects whether or not it receives the third ready signal from the slave interface card in a set period. If the third ready signal is received, it enters to step 325 and the master interface card sends confirm signals to all the slave interface cards. If the third ready signal is not received, it returns to the master test table as shown in step 321. This means some errors occur in the testing procedure.

[0029] As shown in step 360, the slave interface card detects whether it receives the instruct signal from the master interface card to decide either to exiting or ~~resuming~~ to resume. If the instruct signal is not received in a set period, it returns to the slave test table in step 361. If the slave interface card receives the instruct signal of ~~existing~~ exiting, it returns to slave test table. If the slave interface card receives the instruct signal of resuming, it returns to the S1 node of FIG. 2B and resumes the test.

[0030] According to the invention, all the acts of returning to the master/slave test table are monitored. However, it is a normal situation if the act is decided by the test worker, such as step 340 and step 370, and it does not mean ~~there is any~~ an error has occurred. Otherwise, it is an abnormal situation if the act is not decided by the test

worker. It means some error occurs and the test worker is able to control and solve the problem by debugging.

[0032] (1) It can fully control the buffer and monitor the memory since the test method is not performed under the driver structure in Microsoft's ~~Windows Operating System~~ WINDOWS operating system . Besides, it is convenient to test in a load flow environment by using this method.

[0035] (4) The method can be performed in a multiple system and ~~it~~ there is no need to be restricted ~~in~~ to use with the ~~Windows Operating System~~ WINDOWS operating system.